DOCKET NO.
NAI1P017/01.062.01

U.S. PATENT APPLICATION

FOR A

SYSTEM, METHOD AND COMPUTER PROGRAM PRODUCT FOR TESTING PRE-RELEASE ANTI-VIRUS UPDATES

INVENTOR(S): William Alexander McEwan

ASSIGNEE: NETWORKS ASSOCIATES TECHNOLOGY, INC.

KEVIN J. ZILKA
PATENT AGENT
P.O. BOX 721120
SAN JOSE, CA 95172

SYSTEM, METHOD AND COMPUTER PROGRAM PRODUCT FOR TESTING PRE-RELEASE ANTI-VIRUS UPDATES

FIELD OF THE INVENTION

The present invention relates to virus scanning methods, and more particularly to distributing scanning updates for scanning data for viruses and/or harmful content.

BACKGROUND OF THE INVENTION

The generation and spread of computer viruses is a major problem in modern day computing. Generally, a computer virus is a program that is capable of attaching to other programs or sets of computer instructions, replicating itself, and performing unsolicited or malicious actions on a computer system. Generally, computer viruses are designed to spread by attaching to floppy disks or data transmissions between computer users, and are designed to do damage while remaining undetected. The damage done by computer viruses may range from mild interference with a program, such as the display of an unwanted political message in a dialog box, to the complete destruction of data on a user's hard drive. It is estimated that new viruses are created at a rate of over 100 per month.

A variety of programs have been developed to detect and destroy computer viruses. As is known in the art, a common method of detecting viruses is to use a virus scanning engine to scan for known computer viruses in executable files, application macro files, disk boot sectors, etc. Generally, computer viruses are comprised of binary sequences called "virus signatures." Upon the detection or a virus signature by the virus scanning engine, a virus disinfection program may then be used to extract the harmful information from the infected code, thereby disinfecting that code. Common virus scanning software allows for boot-sector scanning upon system bootup, on-demand scanning at the explicit request of the user, and/or on-access scanning of a file when that file is accessed by the operating system or an application.

In order to detect computer viruses, a virus scanning engine is generally provided in conjunction with one or more files called "virus signature files". The virus scanning engine scans a user's computer files via a serial comparison of each file against the virus signature files. Importantly, if the signature of a certain virus is not contained in any of the virus signature files, that virus will not be detected by the virus scanning engine.

Generally speaking, a recent trend is for manufacturers of antivirus applications to update their virus signature files as new viruses are discovered and as cures for these viruses are developed, and to make these updated signature files available to users on a periodic basis (e.g. monthly, quarterly, etc.). For example, an antivirus program manufacturer may post the update file on a bulletin board system, on an FTP (File Transfer Protocol) site, or on a World Wide Web site for downloading by users.

Updates to antivirus applications often must be developed and tested in short time cycles so that customers can be protected for new virus threats. Antivirus applications also must operate as part of an operating system, so the quality of antivirus applications must be high to prevent system failure.

As a result of the rapid nature of development of antivirus application updates, and the wide scale distribution via the Internet, current-testing procedures do not always ensure stability. Unfortunately, various problems can occur when antivirus applications or signature files are updated, i.e. system hangs, system crashes and false alarms (i.e. detecting viruses when no virus exists).

DISCLOSURE OF THE INVENTION

A system, method and computer program product are provided for testing scanner updates. Initially, a full-release scanner update is distributed from a server to a plurality of computers utilizing a network. A pre-release scanner update is also distributed from the server to the computers utilizing the network. The full-release scanner update is executed on the computers for security scanning. Further, the pre-release scanner update is executed on the computers for testing purposes. Results of the testing are transmitted from the computers to the server utilizing the network.

In one embodiment, the full-release scanner update and the pre-release scanner update may be distributed simultaneously. Further, the full-release scanner update and the pre-release scanner update may be distributed together. As an option, the pre-release scanner update may be distributed with virus signature updates. Still yet, the pre-release scanner update may be distributed on a periodic basis. In use, it may be determined whether a pre-release scanner update exists, and the pre-release scanner update may be conditionally distributed from the server if the pre-release scanner update exists. Optionally, the pre-release scanner update may be executed when the computers are idle.

In another embodiment, the results of the testing are transmitted to a quality assurance administrator via the Internet. Further, results of the execution of the full-release scanner update and the pre-release scanner update may be compared. It should be noted that the comparison may occur on the computers. In such case, the results of the comparison are transmitted from the computers to the server utilizing the network. In the alternative, the comparison may be performed by the quality assurance administrator.

In still another embodiment, it may be determined whether a virus is detected by the execution of the pre-release scanner update and not the full-release scanner update based on the comparison. Such virus may be removed, and an associated record of the virus may be stored. The virus may also be reported.

In a similar manner, faults associated with the execution of the pre-release scanner update may be detected. A record of such faults may be transmitted from the computers to the server utilizing the network. Further, the execution of the pre-release scanner update on the computers may be ceased in response to the detection of the faults. Thereafter, the pre-release scanner update may be re-executed on the computers after ceasing the execution. A number of the faults may be counted such that the pre-release scanner update may be conditionally re-executed on the computers based on the number.

In still yet another embodiment, a duration of the execution of the pre-release scanner update may be monitored. Further, a record of the duration may be transmitted from the computers to the server utilizing the network.

By this design, the results of executing one or more pre-release scanner updates may be used to modify the pre-release scanner update before releasing the pre-release scanner update as a full-release scanner update.

Brief Description of the Drawings

Figure **1** illustrates a network architecture, in accordance with one embodiment.

Figure **2** shows a representative hardware environment that may be associated with the remote server and/or target computers of Figure **1**, in accordance with one embodiment.

Figure **3** illustrates an overview of a method for testing scanner updates, in accordance with one embodiment.

Figures **4** through **6** illustrate a method for testing scanner updates, in accordance with another embodiment.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

Figure **1** illustrates a network architecture **100**, in accordance with one embodiment. As shown, a remote server **112** is provided which is coupled to a network **102**. In the context of the present network architecture **100**, the network **102** may take any form including, but not limited to a local area network (LAN), a wide area network (WAN) such as the Internet, etc. It should be noted that various other networks **102** may also be included.

Also provided is at least one target computer **114** coupled to the network **102**. Such target computers **114** may include a web server, desktop computer, lap-top computer, hand-held computer, printer or any other type of hardware/software. For reasons that will soon become apparent, each of the target computers **114** may include a scanner for performing virus and/or content scanning. In particular, each scanner may serve to scan the target computer **114** for malicious programs such as viruses, worms, and Trojan horses. Further, each scanner may serve to filter content at the associated target computer **114** to enforce operational policies [i.e. detecting harassing or pornographic content, junk e-mails, misinformation (virus hoaxes), etc.].

The remote source **112** includes a plurality of scanner update databases **120**. In the context of the present description, a scanner update in the scanner update databases **120** may include virus signatures, rule sets, or any other component of the scanners on the target computers **114** which may be updated for improvement purposes. Accordingly, the scanner updates may be used for virus, content or another type of scanning.

The scanner update databases **120** include a full-release scanner update database **122** and a pre-release scanner update database **124**. The full-release scanner update database **122** includes a plurality of full-release scanner updates that are tested and ready for distribution to the target computers **114** for updating the scanners thereon. On the other hand, the pre-release scanner update database **124** includes pre-release scanner updates which are not necessarily fully tested and ready for distribution. In the context of the present description, the pre-release scanner updates may include any scanner updates less ready for distribution with respect to the full-release scanner updates.

Figure **2** shows a representative hardware environment that may be associated with the remote server **112** and/or target computers **114** of Figure **1**, in accordance with one embodiment. Such figure illustrates a typical hardware configuration of a workstation in accordance with a preferred embodiment having a central processing unit **210**, such as a microprocessor, and a number of other units interconnected via a system bus **212**.

The workstation shown in Figure **2** includes a Random Access Memory (RAM) **214**, Read Only Memory (ROM) **216**, an I/O adapter **218** for connecting peripheral devices such as disk storage units **220** to the bus **212**, a user interface adapter **222** for connecting a keyboard **224**, a mouse **226**, a speaker **228**, a microphone **232**, and/or other user interface devices such as a touch screen (not shown) to the bus **212**, communication adapter **234** for connecting the workstation to a communication network **235** (e.g., a data processing network) and a display adapter **236** for connecting the bus **212** to a display device **238**.

The workstation may have resident thereon an operating system such as the Microsoft Windows NT or Windows/95 Operating System (OS), the IBM OS/2 operating system, the MAC OS, or UNIX operating system. It will be appreciated that a preferred embodiment may also be implemented on platforms and operating systems other than those mentioned. A preferred embodiment may be written using JAVA, C, and/or C++ language, or other programming languages, along with an object oriented programming methodology. Object oriented programming (OOP) has become increasingly used to develop complex applications.

Figure **3** illustrates an overview of a method **300** for testing scanner updates, in accordance with one embodiment. Initially, in operation **1**, a full-release scanner update from the full-release scanner update database **120** is distributed from the server **112** to a plurality of computers **114** utilizing the network **102**. As shown, a pre-release scanner update from the pre-release scanner update database **124** is also distributed from the server **112** to the computers **114**.

Next, in operation **2**, the full-release scanner update is executed on the computers **114** for security scanning. Further, the pre-release scanner update is executed on the computers **114** for testing purposes.

As shown in operation **3**, results of the testing are transmitted from the computers **114** to the server **112** utilizing the network **102**. By this design, the results of one or more pre-release scanner updates may be used by a quality assurance administrator to modify the pre-release scanner update before releasing the pre-release scanner update as a full-release scanner update. See operations **4** and **5** of Figure **3**.

Figures **4** through **6** illustrate a method **400** for testing scanner updates, in accordance with another embodiment. As mentioned hereinabove, a full-release scanner update database **120** is distributed from the server **112** to a plurality of computers **114** utilizing the network **102**. Note operation **402**. Further, in operation **404**, a pre-release scanner update from the pre-release scanner update database **124** is also distributed from the server **112** to the computers **114**. When distributed, the full-release scanner update replaces an old update, and the pre-release scanner update is stored on the computers **114** for testing in a manner that will soon be set forth.

In one embodiment, the full-release scanner update and the pre-release scanner update may be distributed simultaneously. Further, the full-release scanner update and the pre-release scanner update may be distributed together in the same file and/or data transmission. Still yet, the scanner updates may be distributed on a periodic basis, i.e. daily, weekly, monthly, etc.

As an option, the pre-release scanner update may be conditionally distributed from the server **112** based on whether the pre-release scanner update exists. For example, it may be determined whether a pre-release scanner update exists, and the pre-release scanner update may be conditionally distributed from the server **112** if the pre-release scanner update exists. In such embodiment, the pre-release scanner update may be distributed with the full-release scanner update based on some predetermined development schedule.

In operation **406**, the full-release scanner update is executed with the scanner on the computers **114** for conventional security scanning. Results of the execution of the full-release scanner update are stored for analysis at a later time.

It is then determined whether a particular computer **114** is idle in decision **408**. In the context of the present description, the computer **114** may be considered idle when more than a predetermined amount of resources is available. As an option, the computer **114** may be considered idle when a user is not using the computer **114**, or no applications are currently being executed.

Once the computer is considered idle, the pre-release scanner update is executed with the scanner on the computer **114** for testing the pre-release scanner update. Note operation **410**. It should be noted that the operations associated with the full-release scanner update are recorded in a queued so that, during idle time, the pre-release scanner update may run the same operations as performed by the full-release scanner update for comparison of the results.

Results of the execution of the pre-release scanner update and the full-release scanner update are then compared in operation **412**. It should be noted that the comparison may occur on the computers **114**. In the alternative, the comparison may be performed by the quality assurance administrator at the server **112**.

In particular, it may be determined whether a virus is detected by the execution of the pre-release scanner update and not the full-release scanner update based on the comparison. Note operation **414**. If a virus is detected by the pre-release scanner update which was not detected by the full-release update, it is possible that such detection is a false alarm. A test for such situation may be carried out, as will soon become apparent.

With reference now to Figure **5**, it is shown that it is determined in decision **500** whether the virus is detected. If it is, the virus is removed in operation **502** using any well known repair routine. Further, an associated record of the virus may be stored at the computer **114** in operation **504**. Such record may include a virus signature, time of detection, or any other identifying information.

A duration of the execution of the pre-release scanner update is monitored during execution. If a delay in such execution exceeds a predetermined amount, as determined in decision **508**, a record of the duration may be stored at the computer **114** in operation **510**. In one embodiment, decision **508** may gauge the delay based on whether the time taken to scan is proportionally longer with respect to the full-release scanner update, or, in other words, the pre-release scanner update is considerably slower. As an option, a certain percentage range (i.e. 10%-X%) of increased delay may trigger decision **508.** Further, the record of duration may be in terms of a difference with respect to the execution of the full-release scanner update.

Next, in decision **512**, the execution of the pre-release scanner update on the computer **114** is monitored for a failure to complete a designated scan. Such event may be triggered upon the execution of the pre-release scanner update exceeding the percentage range mentioned hereinabove. In response to such unacceptable delay, the execution of the pre-release scanner update may be ceased in operation **514**. Further, the failure may be recorded in operation **516**, and the pre-release scanner update re-executed in operation **518**.

In a similar manner, crashes associated with the execution of the pre-release scanner update may be detected in decision **519**. In one embodiment, a crash may refer to any instance where an error is returned, or any other indication that the pre-release scanner update or computer **114** has stopped working. If detected, the crash of the pre-release scanner update on the computer **114** may be recorded. See operation **520**. Thereafter, the pre-release scanner update may be re-executed in operation **522.**

As mentioned earlier during reference to operation **414**, if a virus is detected by the pre-release scanner update which was not detected by the full-release update, it is possible that such detection is a false alarm. A test for such situation may be carried out in decision **524**. This may be accomplished by any well known analysis of viruses and the legitimacy thereof. If the virus detection is determined to be a false alarm, a record of such is stored in operation **526**.

For reasons that will soon become apparent, a number of the foregoing instances of faults (i.e. delays, failures, crashes, false alarms, etc.) may be counted for determining whether the number of such instances exceed a predetermined amount. Note decision **528**. In one embodiment, the predetermined amount may refer to a predetermined number of instances in a certain timeframe. Such criteria may be set to prevent the computer **114** from stalling beyond a predetermined threshold. If such predetermined amount is not yet reached, the pre-release scanner update may be continuously executed and re-executed on the computer **114**. If, however, such predetermined amount has been reached, the pre-release scanner update may be terminated in operation **530**.

Next, the method **400** is continued by transmitting results to the server **112** in operation **606** of Figure **6**. Ideally, a record of such viruses, failures, crashes, false alarms and delays found are transmitted from the computers to the server utilizing the network.

By this design, the results of one or more pre-release scanner updates may be used by a quality assurance administrator to modify the pre-release scanner update in operation **608**. Specifically, the pre-release scanner update may be modified to prevent the faults before releasing the pre-release scanner update as a full-release scanner update in operation **610**.

The present embodiment thus provides a distributed testing framework that can be created as part of the scanning software product which allows for transparent testing of pre-release updates. This may be done at idle time on computers **114** by using redundancy checking against a currently full-release update. To this end, the present testing framework provides a safe process in which pre-release updates can be tested automatically without affecting the computers **114**.

While various embodiments have been described above, it should be understood that they have been presented by way of example only, and not limitation. Thus, the breadth and scope of a preferred embodiment should not be limited by any of the above-described exemplary embodiments, but should be defined only in accordance with the following claims and their equivalents.